My project is a citation graph analysis tool that helps researchers explore the relationships between academic papers in the citation network from Arxiv on high energy physics papers.The code works by representing the citations between papers as a directed graph, where each node represents a paper and each edge represents a citation from one paper to another. Using the graph, it does two things. First, it can find the shortest path between any two papers in the graph using a breadth-first search algorithm. This would help researchers understand how papers and their concepts are related to each other and identify potential research connections. Second, it can identify the top five most cited papers in the graph. This can help researchers identify important or influential papers in the field of high-energy physics.

The first module is the paper.rs module that reads the data from the txt file and creates a hashmap that contains the ID number of each paper. The second module is the citation_graph.rs module which defines a CitationEdge struct that represents a directed edge in a citation graph. It has two fields, parent_id and child_id, which are the IDs of the parent and child papers in the graph, respectively. The CitationGraph struct represents a citation graph and has a field for edges, which is a vector of CitationEdge instances. It also has two methods: from_edges which creates a new CitationGraph instance from a vector of CitationEdge instances and from_papers which creates a new CitationGraph instance from a HashMap of Paper instances. The from_papers method generates a complete graph where each paper is connected to every other paper in the hashmap. It does this by iterating over all possible pairs of papers, and creating a CitationEdge between each pair. Finally, it creates a CitationGraph from the generated edges and returns it.

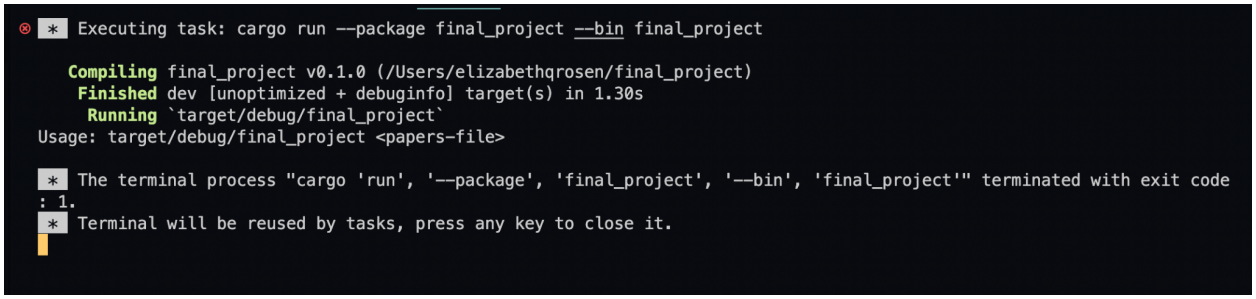
The analysis.rs module performs six degrees of separation analysis using breadth first search and also finds the top five most cited papers. The six_degrees_of_separation function implements the "six degrees of separation" algorithm to find the shortest path between two papers in a citation graph. It takes four parameters: start_paper_id, end_paper_id, citation_graph, and _papers. First, the function constructs a mapping of paper IDs to their adjacent papers, by iterating over the edges in the CitationGraph and inserting the child paper ID into the set of adjacent papers for the parent paper ID in the adjacency_map HashMap. Next, the function performs a breadth-first search to find the shortest path from the start_paper_id to the end_paper_id. It uses a HashSet to keep track of which papers have been visited, a VecDeque to store the order in which papers should be visited next, and a HashMap to store the previous paper ID for each visited paper ID. If the end_paper_id is found during the search, the function constructs and returns the path from the start_paper_id to the end_paper_id by following the prev_paper map in reverse order. If no path is found after searching all reachable papers, the function returns `None`.

The find_most_cited_papers function takes a CitationGraph object and returns a vector of the top 5 papers with the most citations. It works by iterating over the edges in the CitationGraph and counting the number of times each paper ID appears as a child paper ID. It then sorts the paper IDs by the number of citations in descending order using a stable sorting

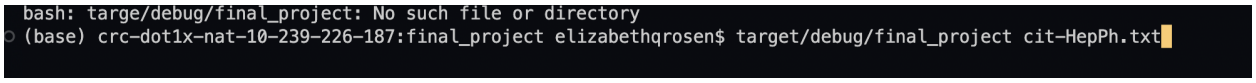
algorithm (`sort_by_cached_key`), and takes the top 5 paper IDs to return as the most cited papers.

The main function utilizes the functions defined in these modules to actually perform the analysis.

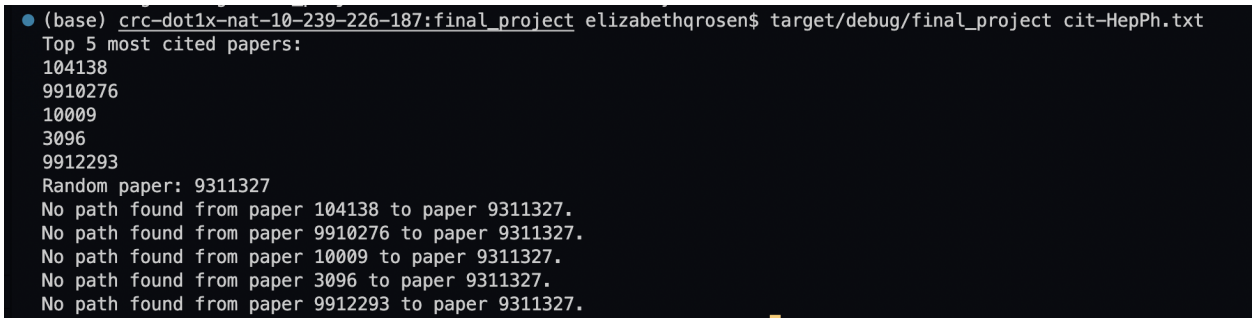
To run the code, press the run button above the main function, then when prompted with the following:



Press any key, and then type the following: target/debug/final_project cit-HepPh.txt like this:



Then wait for the output which should look like this:



The output can take a while to create, and I was unable to find a way to increase the efficiency of the code.

Resources:
https://doc.rust-lang.org/stable/std/collections/struct.VecDeque.html
https://programming-idioms.org/idiom/19/reverse-a-list/444/rust
https://docs.rs/radsort/latest/radsort/fn.sort_by_cached_key.html